Review Sheet for Final Exam
APPM/MATH 4650 Fall '20 Numerical Analysis

Instructor: Prof. Becker

Note: the final exam is cumulative, but will focus a bit more on recent material (ch 5 and 6) not tested on midterm 2. This review sheet does not cover the entire class (it only covers ch 5 and 6), so please review the midterm 1 and midterm 2 review sheets.

Chapter 5: IVPs and ODEs

1. Why don't we see higher-order Taylor methods in software much?
2. How many "steps" is RK4?
3. What's the big deal about an "embedded" RK formula?
4. What are the pros/cons of one-step methods vs multi-step methods?
5. How are Adams methods derived?
6. What's the main difference between Adams-Bashforth and Adams-Moulton?
7. What does it mean to say a method is implicit or explicit?
8. What's the idea of a predictor-corrector method?
9. How might one solve for the update step of an implicit method?
10. How are backward differentiation (BD) methods derived?
11. What's so special about Adams and BD methods? Can't we create more?
12. What types of higher-order ODEs can be recast as a system of first-order ODEs?
13. How does the book define "stability" of a numerical IVP method?
14. What does it mean to say a method is "consistent"?
15. What's the local truncation error (LTE)?
16. If the LTE is $O(h^3)$, what kind of error would we expect the global error to be?
17. For finite difference methods for derivatives, we worried a lot about roundoff error. ODEs involve derivatives. Are we really worried about roundoff error for ODE solvers?
18. What does "convergence" mean?
19. How are stability, convergence and consistency related?
20. How do we check for the stability of a one-step method?
21. How do we check for the stability of a multi-step method?
22. Is "absolute stability" just "stability" with absolute values? What is it?
23. What does it mean to have a "stiff" equation? Why is absolute stability discussed in this chapter?
24. Are some methods absolutely stable while others are not?
25. What is the region of stability? How do we calculate it?

Chapter 6: IVPs and ODEs

1. What kind of linear equations did we mostly discuss solving (over-determined? under-determined? square? consistent? singular?

2. True or false: if A is a square matrix, then we can solve $Ax = b$ iff A is invertible

3. Let

$$A = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{bmatrix}, \quad B = \begin{bmatrix} — & \mathbf{b}_1^T & — \\ — & \mathbf{b}_2^T & — \\ — & \mathbf{b}_3^T & — \end{bmatrix},$$

   i.e., A has columns $\mathbf{a}_i$ and B has rows $\mathbf{b}_j^T$. Write down an expression for AB in terms of $\mathbf{a}_i$ and $\mathbf{b}_j$.

4. For A and B defined as above, what is the $(i,j)^{\text{th}}$ entry of BA?

5. What's the complexity of matrix multiplication of a $n \times n$ matrix times a vector?

6. What's the complexity of matrix multiplication of a $n \times n$ matrix times another matrix of the same size?

7. What's the complexity of performing Gaussian elimination on a $n \times n$ matrix times?

8. What's the complexity of solving a linear system of equations if the matrix has a triangular structure?

9. How do we use the LU factorization to solve a system of equations?

10. How do we use a pivoted LU factorization to solve a system of equations?

11. True/false: Numerical methods only exchange rows in the LU factorization if they have a 0 pivot

12. Do we need to pivot when computing an LDL factorization? when computing a Cholesky factorization?

13. What is the standard condition number of matrix?

14. Why would someone write a blocked LU factorization code?

15. Let A be a $n \times n$ matrix. If your code calls $A\mathbf{b}_i$ for $i = 1, 2, \ldots, n$, this is $O(n^3)$ flops. If your code instead forms the matrix B with columns $\mathbf{b}_i$ and then multiplies AB, this is also $O(n^3)$. Is there a difference in speed in practice? Why?

16. Your friend Tara needs to solve $A\mathbf{x}_i = \mathbf{b}_i$ for $i = 1, 2, \ldots, 1000$. Rather than solve the systems independently, they cleverly precompute $B = A^{-1}$ once, and then can solve for $\mathbf{x}_i$ via $\mathbf{x}_i = B\mathbf{b}_i$ efficiently. What's a better way?